The ITOS Command Subsystem

ITOS User's Manual
$Date: 2006/07/13 15:36:34 $

The ITOS Command Subsystem

1 Command Overview

The command processing system supports the Consultative Committee for Space Data Systems (CCSDS) Command Operation Procedure (COP), COP-1 Enhanced Service. COP-1 fully specifies the closed-loop protocol executed between the sending (ground) and receiving (spacecraft) entities. The COP consists of a pair of synchronized procedures: a Frame Operation Procedure (FOP) that executes within the sending entity; and a Frame Acceptance and Reporting Mechanism (FARM) that executes withing the receiving entity. The FOP transmits telecommand transfer frames to the FARM. The FARM returns telemetered Command Link Control Words (CLCW's) within the telemetry transfer frame trailers to the FOP which provides return status about the acceptance of the frames by the spacecraft.

COP-1 operates on the principle of sequential frame acceptance and retransmission, with frame sequence numbering. The FOP initiates the transmission of TC Frames whose sequence numbers are arranged in upcounting sequential order. The FARM only accepts frames if their sequence numbers match the expected upcounting order. As soon as a sequence error is encountered, the FARM rejects all subsequent frames whose sequence numbers do not match the expected order. The FOP monitors the CLCW to determine if frames are being rejected, and if so, backs up and retransmits the series of frames beginning with the frame whose sequence number matches the number which the FARM is expecting. See the CCSDS 202.0-B-1 Blue Book for complete specification of the COP-1 protocol.

2 Enabling Commands

The command subsystem must be activated before the spacecraft can be commanded. This is done with the ENABLE CMD directive. Only one user, which should be the Test Conductor on the Test Conductor Workstation, may have commands enabled at any given time. In the unlikely event that the Test Conductor Workstation experiences a hardware failure, commands may be enabled on another workstation (referred to as a Subsystem GSE) which could takeover as the new Test Conductor Workstation.

3 Types of Commands

3.1 Command Mode Configuration Commands

The MODE directive (see ⟨undefined⟩ [MODE], page ⟨undefined⟩) allows the user to configure the various commanding modes that control the manner in which spacecraft commands are processed and transmitted by ITOS.

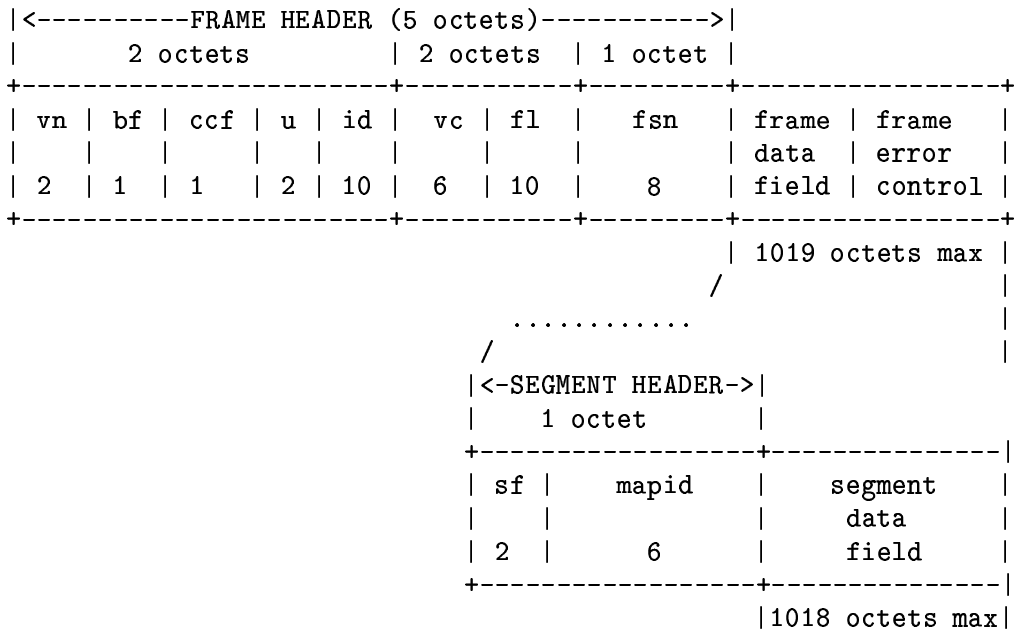
There are two commanding modes, one-step and two-step, which affect the manner in which spacecraft commands are transmitted to the spacecraft. While in one-step commanding mode, spacecraft commands entered by the user are formatted into telecommand transfer frames and immediately transmitted to the spacecraft. In two-step mode, commands are formatted and stored in the command buffer where they remain until a request is received by the user to either transmit the commands or clear them from the buffer. For details on the command buffer and instructions for viewing its contents, see See Chapter 7 [Command Buffer], page 12. After a command is successfully transmitted to the spacecraft, it is placed in the Sent-Queue pending acknowledgement by the spacecraft. The Sent-Queue page display provides a history of the most recently transmitted commands as well as information about currently pending commands. For details on the Sent-Queue, see Chapter 8 [Sent Queue], page 13.

3.2 Spacecraft Commands

Spacecraft commands are used to control individual components of the spacecraft. They may be issued by the user in either mnemonic format using the CMD directive (see ⟨undefined⟩ [CMD], page ⟨undefined⟩), or raw format using the RAW (see ⟨undefined⟩ [RAW], page ⟨undefined⟩) and RAWTF (see ⟨undefined⟩ [RAWTF], page ⟨undefined⟩) directives. Commands entered in mnemonic form are formatted into Telecommand Packets which are encapsulated in Telecommand Transfer Frames as defined in the Database described in the Telemetry and Command Handbook.

The Telecommand Transfer Frame format is:

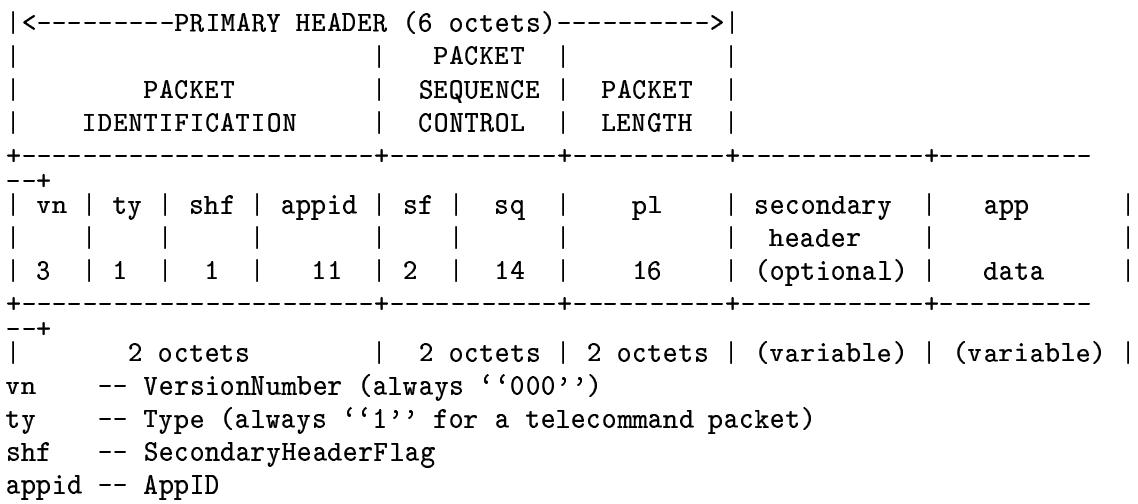
```
From CCSDS 202.0-B-2: _Telecommand_Part_2_-_Data_Routing_Service_,
November 1992, pages 3-3 and 4-3:
```



vn - VersionNumber -- constant 0 (0b00)
bf - BypassFlag -- normally 0, 1 indicates a bypass command
ccf - ControlCommandFlag -- normally 0, 1 indicates a control command
u - reserved spares -- constant 0
id - SpacecraftId
vc - VirtualChannel
fl - FrameLength -- ``C'', where ``C = (Total Number of Octets) - 1''
fsn - FrameSequenceNumber
sf - sequence flags -- normally ``11'' (no segmentation)
mapid - Multiplexer Access Point (MAP) Identifier

The Telecommand Packet format is:

From CCSDS 203.0-B-1: _Telecommand_Part_3_-_Data_Management_Service_, January 1987, page 5-2:



vn -- VersionNumber (always ``000'')
ty -- Type (always ``1'' for a telecommand packet)
shf -- SecondaryHeaderFlag
appid -- AppID

```
sf    -- SequenceFlags
sq    -- SequenceCount
pl    -- PacketLength ``C'' where ``C = (number of octets) - 1''
```

All commands received by the spacecraft undergo a basic set of "Frame Validation Checks". If the Bypass Flag in the TC Transfer Frame Header is not set, each Frame is additionally tested against the Frame Acceptance Checks. Frames that are transmitted in the normal "Acceptance" mode undergo both the Validation and Acceptance Checks are called Type-A Frames. Frames that have their Bypass Flag set cause the normal Frame Acceptance Checks of the FARM to by bypassed. These Frames are called Type-B or Bypass Frames.

The FOP can be configured to send commands as Type-B Frames, by using the BYPASS ON directive (see ⟨undefined⟩ [BYPASS], page ⟨undefined⟩) to enter bypass commanding mode. All commands entered in this mode will have their Bypass Flag set to 1. The Bypass Mode can be used whenever there is no data being telemetered by the spacecraft, thus making command verification impossible.

3.2.1 Command Mnemonics

Commands that are defined in the database may be entered in mnemonic form using the CMD directive (see ⟨undefined⟩ [CMD], page ⟨undefined⟩). Some commands may have associated submnemonics which further define the action of the commands. Submnemonics may be fixed, variable, or invisible. Fixed submnemonics, as the name implies, may only take on fixed values which are predetermined and defined in the database (eg. `ON`, `OFF`). Variable submnemonics may take on any value within a specified range of values (eg. `ADDRESS=h'7FF'`). All Fixed and Variable submnemonics must be entered in the command line. Invisible submnemonics are fixed submnemonics that can only take on one predefined value. Since the value of an invisible submnemonic can never change, it does not have to be specified on the command line, but will automatically be included in the formatted command which is transmitted to the spacecraft. Some examples of a command which has both fixed and variable submnemonics are:

```
/CMD MUDOWNSETUP MUESOURCE, BIPHASE, TLM4K, SELECTVOLT=5
/CMD MUDOWNSETUP IDPUSOURCE, NRZL, TLM1500K, SELECTVOLT=10
```

3.2.2 Raw Commands

The user may enter commands in raw hexidecimal format at either the CCSDS Source Packet or Teleocmmand Transfer Frame level. Raw commands may be used to send commands that are not defined in the database or to allow negative command testing of the spacecraft FARM by setting fields in the TC Transfer Frame header. The RAW directive (see ⟨undefined⟩ [RAW], page ⟨undefined⟩) is used to send raw Telecommand Source Packets, and the RAWTF directive see ⟨undefined⟩ [RAWTF], page ⟨undefined⟩ is used to send raw telecommand Transfer Frames. For raw commands entered at the Source Packet level, the command subsystem will automatically format the Packet into a Transfer Frame before transmitting the command to the spacecraft.

3.2.3 Hazardous Command Screening

Depending on the current state of the spacecraft, the functions that some commands perform may be considered hazardous. Commands that may be hazardous are defined as such in the database. The criticality of all spacecraft commands, except when entered in raw format, is screened. If the database indicates that a particular command is hazardous, the user will be notified and prompted to allow or cancel the command. When commanding in two-step mode, if any command in the block of commands entered are hazardous, upon issuing the SEND directive (see ⟨undefined⟩ [SEND], page ⟨undefined⟩), the user will be prompted to allow or cancel the transmission of the entire block of commands. **This is not currently implemented and no prompt will occur.**

3.3 Control Commands

Control commands are used to control and configure the FOP and FARM. FOP control commands are executed by the FOP, and FARM control commands are transmitted to the spacecraft for execution by the FARM.

3.3.1 FARM Control Commands

FARM control commands contain data which sets the parameters of the FARM to the proper configuration to accept telecommand data. All FARM control commands have their Bypass Flag and Control Command Flag set to the value `1`. The FARM control commands supported by ITOS at this time are UNLOCK, SET NEXT EXPECTED FRAME SEQUENCE NUMBER and SET NEXT EXPECTED FRAME SEQUENCE NUMBER TO ZERO. These control commands are sent via the UNLOCK(see ⟨undefined⟩ [UNLOCK], page ⟨undefined⟩) SETVR(see ⟨undefined⟩ [SETVR], page ⟨undefined⟩) and RESET (see ⟨undefined⟩ [RESET], page ⟨undefined⟩) directives, respectively.

3.3.2 FOP Control Commands

FOP control commands are used to configure the FOP for the desired mode of operation. The FOP can be configured to turn off command verification using the `BLIND` (see ⟨undefined⟩ [BLIND], page ⟨undefined⟩) or the `MODE` (see ⟨undefined⟩ [MODE], page ⟨undefined⟩) directive. The automatic retransmission of commands can be configured by the `RETRLIM` (see ⟨undefined⟩ [RETRLIM], page ⟨undefined⟩) and `RETRY` (see ⟨undefined⟩ [RETRY], page ⟨undefined⟩) directives. The CLCW timeout parameter can be configured by the `TIMEOUT` (see ⟨undefined⟩ [TIMEOUT], page ⟨undefined⟩) or `MODE` directive. And the commands in the SENT QUEUE pending acknowledgement by the FOP can be forcibly removed using the `PURGE` (see ⟨undefined⟩ [PURGE], page ⟨undefined⟩) directive.

4 Spacecraft Command Verification

The FOP verifies that the commands are received correctly by the spacecraft, but does not verify that the commands are actually executed (otherwise known as end-item verification). The user can achieve end-item verification using STOL procedures which can issue a spacecraft command and then wait for a specified telemetry point to change to an expected value.

Command verification is performed according to the CCSDS COP-1 protocol. Type-A commands sent to the spacecraft must be received in strict upcounting sequential order. In order to maintain this strict order, each spacecraft command is formatted into a transfer frame, assigned a sequential frame sequence number, and transmitted to the spacecraft. The spacecraft looks at the frame sequence numbers of the transfer frames and makes sure they are received in sequential order. The spacecraft then determines the next expected frame sequence number (NE) that is should receive and inserts this information in the Command Link Control Word (CLCW) of a transfer frame to be telemetered to the ground. The ground maintains a separate frame sequence number counter which is used to number the transfer frames.

If a transfer frame is received out of sequence, the spacecraft sets a flag in the CLCW indicating that a command retransmission beginning at the NE will be necessary. If autoretransmission mode is in effect, the failed commands will be retransmitted until they are either verified or the number of retransmissions have exceeded the maximum allowable limit.

Because the Telecommand Transfer Frame Sequence Number field in the Frame Header is only 8 bits long, the frame sequence number will wrap-around every 256 frames. There is a mechanism used by the FARM, called the FARM Sliding Window, which is used to prevent a complete "wrap-around" of the transmitted frame sequence numbers. If a frame is received with a Frame Sequence Number greater than the limit established by the FARM Sliding Window, the FARM will set the Lockout flag in the CLCW indicating that it is in LOCKOUT mode. Once in LOCKOUT mode, the FARM will not accept any Type-A frames until the channel is unlocked via the UNLOCK control command (see ⟨undefined⟩ [UNLOCK], page ⟨undefined⟩).

If the CLCW is not updated within the maximum allowable time after transmitting a command, a TIMEOUT condition will occur and the command will fail verification. This CLCW timeout period is programmable (see ⟨undefined⟩ [TIMEOUT], page ⟨undefined⟩).

The commands in the Sent-Queue pending acknowlegement whose frame sequence numbers are less than the spacecraft's next expected frame sequence number are positively verified and are dequeued from the Sent-Queue.

4.1 Automatic Command Retransmission

When a command fails verification, the FOP can be configured to automatically retransmit the command up to a specified number of times until either the command is verified, or the maximum number of retries has been exceeded. Turning on and off automatic retransmission of commands and setting the maximum number of allowable retries is done via

the `RETRLIM` directive (see ⟨undefined⟩ [RETRLIM], page ⟨undefined⟩). Once the maximum number of retries have been reached, the user will be notified of command failure. At this point, the user may manually retransmit the failed command(s) by issuing the `RETRY` directive (see ⟨undefined⟩ [RETRY], page ⟨undefined⟩) , or may remove the failed command from the SENT QUEUE using the `PURGE` directive (see ⟨undefined⟩ [PURGE], page ⟨undefined⟩).

4.2 Setting Timeout Parameter

The CLCW is updated in telemetry at a predictable rate. This rate is determined by such parameters as the round trip propagation delay, the frequency of CLCW sampling, etc. The time between expected updates of the CLCW is configurable by the user via the `TIMEOUT` directive (see ⟨undefined⟩ [TIMEOUT], page ⟨undefined⟩ .

4.3 Single-Step Verification

In single-step verification mode, each Type-A or Type-B spacecraft command must be verified before another command may be issued. Details are TBS.

5 Turning Off Verification

Whenever telemetry is not available from the spacecraft, and, therefore, no CLCW information is being received, command verification may be turned off by going into the Blind Commanding Mode using the `BLIND` directive (see ⟨undefined⟩ [BLIND], page ⟨undefined⟩). In this mode, commands will automatically be purged from the SENT QUEUE.

6 FOP States

Depending on the occurrence of particular events, the FOP will be in one of the following seven states:

'ACTIVE' Fop in nominal status. Ready to accept frames and transmit them.

'WAIT WITH NO RETRANSMIT PENDING'
A CLCW report has indicated that the FARM has no buffer space available. No frames will be transmitted until the condition is cleared.

'WAIT WITH RETRANSMIT PENDING'
Same as above, but in addition, CLCW reports indicate that one or more frames must be retransmitted.

'FARM IN LOCKOUT'
CLCW reports indicate the FARM has detected an anomaly in the functioning of the COP and has entered the LOCKOUT state. Type-B commands may still be transmitted, but no Type-A frames will be transmitted until the condition is cleared.

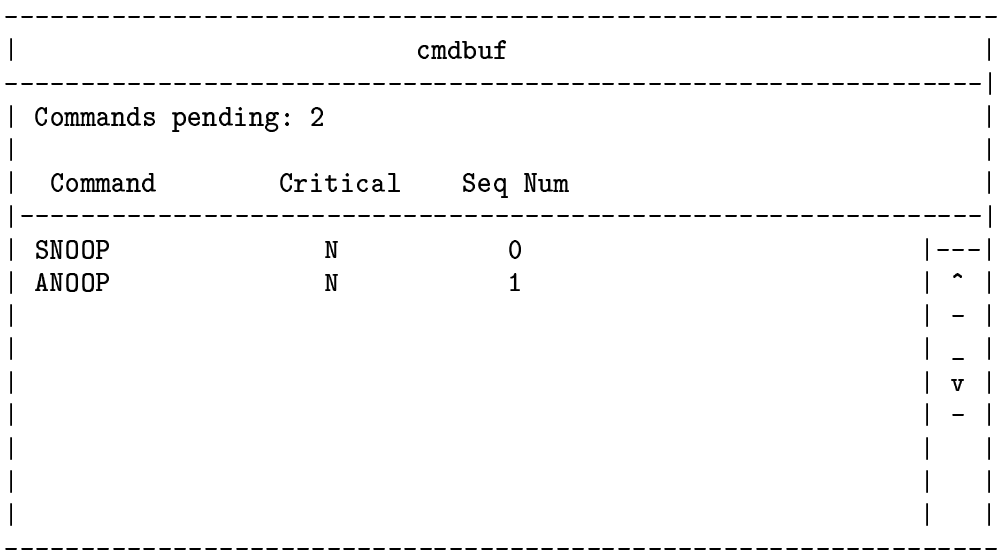
'BLIND COMMANDING'
Commanding without CLCW's.

'INITIAL & FINAL'
State of the FOP when the hardware/software that implements the FOP is started up. FOP is waiting for configuration data. The FOP returns to this state following a commanding session when the channel is closed.

'WAIT TO CLOSE'
Waiting for CLCW reporting to stop following a CLOSE command.

7 Command Buffer

The command buffer is used to store up to 64 commands entered in two-step commanding mode. The commands remain in the buffer until the `SEND` directive (see ⟨undefined⟩ [SEND], page ⟨undefined⟩) is issued. If any command in the command buffer is hazardous as identified in the database, the user will be prompted to allow or cancel the transmission of all of the commands in the buffer. Once transmitted to the spacecraft, the commands will be placed in the SENT QUEUE pending acknowledgment by the spacecraft. Before the `SEND` directive is issued, commands may be cleared from the buffer at any time using the `CLEAR` directive (see ⟨undefined⟩ [CLEAR], page ⟨undefined⟩). The command buffer may be viewed using the `PAGE CMDBUF` directive (see ⟨undefined⟩ [PAGE], page ⟨undefined⟩). To see the raw hexidecimal representation of the commands in the buffer, the user may click on a particular command with the mouse and a pop-up window will appear showing detailed information about the command. An example of the command buffer page is below:



8 Sent Queue

The SENT QUEUE is used to store commands which have been transmitted to the spacecraft and are pending acknowledgment. The SENT QUEUE is also used to provide a history of the most recent 127 commands that have been transmitted to the spacecraft. Commands that fail verification may be purged from the SENT QUEUE by issuing the `PURGE` directive (see ⟨undefined⟩ [PURGE], page ⟨undefined⟩).

The information contained in the SENT QUEUE for each command includes the name of the command, its frame sequence number, the number of times it has been retransmitted, the time it was sent, and its verification status. If a command fails verification, the status will display `FAIL`, if a command passes verification, the status will display `VER`. Because Type-B (bypass) commands cannot be properly verified through telemetry, their status will always be shown as `BYPASS`.

The contents of the SENT QUEUE may be viewed using the `PAGE SENTQ` directive. To see the raw hexadecimal representation of the commands in the buffer, the user may click on a particular command displayed on the SENT QUEUE page with the mouse and a pop-up window will appear showing detailed information about the command. The following is an example of the SENT QUEUE page.

```
------------------------------------------------------------------
|                           sentq                                |
|----------------------------------------------------------------|
| Commands pending: 0   Queue status: EMPTY                      |
|                                                                |
|  Command   Seq Num   Retries   Time sent           Status      |
|----------------------------------------------------------------|
| CTBLDUMP      4          0     93-236-20:13:08.0   VER     |    |
| LOAD          3          0     93-236-20:09:43.0   VER     | -  |
| LOAD          2          0     93-236-20:09:41.0   VER     | ^  |
| LOAD          1          0     93-236-20:09:39.0   VER     | _  |
| ANOOP         0          0     93-236-20:09:28.0   VER     | -  |
| RESET         0          0     93-236-20:09:20.0   BYPASS  | v  |
| SNOOP         5          0     93-236-20:09:06.0   FAIL    |----|
|                                                            |    |
|                                                            |    |
------------------------------------------------------------------
```

9 Image Loads

ITOS provides a *load* capability for sending to the spacecraft a file of data in a stream of telecommands. Loads are initiated by the STOL `load` and `loadpkt` directives. (see ⟨undefined⟩ [LOAD], page ⟨undefined⟩)

Processing initiated by the STOL `load` directive requires a properly-formated load file, discussed in detail below. You can create a properly-formatted load file from an arbitrary binary data file using the ITOS `make_load_file` program. The `loadpkt` directive operates on a formatted load file containing already-built telecommand packets.

ITOS load files consist mainly of the data to be loaded, in ASCII hexadecimal notation. Along with the data, though, the load file contains information on what spacecraft commands must be sent to prepare for the load, to carry the load data, and to finish the load; and additional information on command formatting.

The `load` directive processing splits the data contained in a load file into a series of telecommand source packets, and writes these to a *formatted load file*. To turn on command sequence counting in the source packets set the global mnemonic *GBL_LOAD_SEQUENCE* to 1. The default is 0 or off. Then the formatted load file is uplinked to the spacecraft as it would be if given to the `loadpkt` directive, as follows:

Commands are transmitted through the command buffer, 64 commands at a time. The load process waits until each buffer full of commands is verified before loading and sending the next buffer. If the command system is in two-step mode, the load pauses after filling the command buffer with the first set of commands, waiting for the `send` directive. (see ⟨undefined⟩ [SEND], page ⟨undefined⟩) After the `send` is issued, the load continues as in one-step mode and will not pause again for subsequent buffers full of commands

When all command frames for the image load have been verified by the spacecraft, a global mnemonic, *GBL_LOADDONE*, is set to the value "1". An additional global mnemonic, *GBL_LOADFILE*, is also set which indicates the name of the load file just loaded.

9.1 Load Files

Load files look like this:

```
Table Load
WIRE,TB145,98-133:05:40:05,001,I&T,00C8,NOSWAP
/SMTBLSELECT TABLEID=145, SRCZERO, DESTRAM
/SMTBLLOAD OFFSET=H'0'
/SMTBLCOMMIT CKENABLE, CHECKSUM=H'E0C3'
;
;
X385F12B5 ; Timestamp
;
X 00 04 00 02 99 C9 42 6F 11 F6 3F EE 87 0F 74 30;
X E3 A8 3F D5 14 7B 47 AE 7A E1 3F 84 A9 FC D2 F1;
X 62 4D 3F 60 7E FA BC 6A 93 74 3F 58 00 00 00 00;
...
```

```
X 00 00 00 00 00 00 00 00 00 00 00 00 00 00 99 9A;
X 99 99 99 99 3F F1 33 32 38 3F 6B 10 3F 09 33 32;
X 38 3F 6B 10 3F 09 00 0A 00 00;
;
```

In a load file, comments begin with either semi-colon (;) or hash (#) and continue to the end of the line. Blank lines and lines containing only comments are ignored.

The first non-blank, non-comment line is the abstract record; this is copied to the formatted image load file but otherwise ignored. It is intended as a comment to identify the load.

The second line is the mission information line, which consists of several comma-seperated fields:

mission name
: This field is ignored by the ITOS LOAD directive.

image ID
: This field is ignored by the ITOS LOAD directive.

date
: Copied to the formatted image load file but otherwise ignored.

version
: This field is ignored by the ITOS LOAD directive.

source
: This field is ignored by the ITOS LOAD directive.

packet size
: Maximum packet size. When the LOAD directive formats the raw image load file into packets, this is the maximum number of data bytes in each packet.

swap
: Indicates whether or not the LOAD directive should swap bytes when generating the formatted image load file. Byte swapping is only performed if this field has one of the values `SWAPBYTES` or `UI085`. (Note: `UI085` is recognized but deprecated; `SWAPBYTES` should be used instead). Any other value results in no byte swapping; the value `NOSWAP` is preferred.

  Why swap bytes?

  Some spacecraft (TRACE and WIRE, for example) ingest loads as a sequence of 16-bit little-endian words, which is fine if the load is such a sequence. For an arbitrary sequence of bytes, however, this results in each pair of bytes being swapped.

  An example might make this clearer:

  Suppose we want to load a simple eight byte table so that the byte at the load offset plus zero is 0x00, the byte at load offset plus one is 0x11, the byte at load offset plus two is 0x22, and so on. A load file in the 'correct' byte order would contain:

  ```
  X 00 11 22 33 44 55 66 77
  ```

  However, this must get transmitted as

  ```
  1100332255447766
  ```

  in order for the spacecraft to store the data in memory correctly. (Remember, the spacecraft reads this load as the 'words' 1100, 3322, 5544, and 7766. The spacecraft is little-endian, so when it writes these words to RAM they will end up in the proper order, 0011223344556677.)

data size This is an optional field which gives the size in bytes of data items being loaded. It may be set to '1', '2', or '4'. This option controls how the load program sets the *ADDRESS* or *OFFSET* and *NUMBYTES* fields in the load command. It serves as a divisor for those field values, which normally are set with respect to bytes; that is, byte addresses and byte counts, respectively. Set the data size to 2, and the command fields are set with respect to 16-bit words; set it to 4, and they are set with respect to 32-bit words. In other words, if this field is set to 2, the *ADDRESS* field is calculated to be a 16-bit word address, and *NUMBYTES* is the number of 16-bit words in the data field of the load command. (It really represents *numwords* in that case, though its name still is *NUMBYTES*.)

Consider the load:

```
Code load
UVOT,OPER,2000.11.29-13:00:00,001,I&T,002e,NOSWAP,2
/NOSELECT
/UVOTLOAD OFFSET=H'14',mid=4
/NOCOMMIT
X 006d 0001 0000 3800 00e6 85d0 2a5f
X 8320 0906 b122 4820 400e 4800 4009 8520
X dd40 4820 2000 4810 2002 b700 0d00 0d02
X 0d04 85f0 fd00 81ef b122 8510 383b b100
```

for which the load command is defined as follows:

```
cmd|uvotload|+|0x662|0|||||||||"test load command"
fld|uvotload|mid|+|u12||8|||||||"memory bank ID"
fld|uvotload|offset|+|u1234||10|||||||"load address"
fld|uvotload|numbytes|+|u12||14|||||||"count of load data bytes in this cmd"
fld|uvotload|data|+|u12||16|||||||"load data starts here"
```

This results in the following two load commands. Note the indicated fields, which are the 4-byte *OFFSET* and 2-byte *NUMBYTES* fields:

```
                                                vvvvvvvvvvvv
/LOAD: 004E004302C1 1E62C0000037 0000000400000014001700 6D00010000380000
       E685D02A5F83200906B1224820400E480040098520DD4048202000481020
       02B7000D000D02
/LOAD: 004E002503C1 1E62C0000019 000000040000002B00080D0D85F0FD0081EFB1
       228510383BB100                           ^^^^^^^^^^^^
```

The third line is the *select* command. This is either a command as would be entered via STOL or the string `NOSELECT`. If a command was entered, this command will be placed at the beginning of the formatted load so that it is uplinked before the load data.

The fourth line is the *load* command. This is either a command as would be entered via STOL or the string `NOLOAD`. If a command was entered, the load data will be uplinked using this command. Three *load* command fields are treated specially by the load software:

Load commands are required to have a field named *DATA*, of type `U1`. This field serves only to mark the beginning in the packet of the start of the load data, which continues to the end of the packet.

A load command may have a field giving the address to which the portion of the load contained with any given command should be loaded. If the *load* command has a field

named either *ADDRESS* or *OFFSET*, the load program will increment that field according to the *data size* element in the mission information line. If *ADDRESS* or *OFFSET* is specified in the load command in the load file, then it gives the starting address for the load.

A load command also may have a field named *NUMBYTES*, which gives the number of bytes (or words or longwords, according to *data size*) contained in the load command. If this field is defined, the load program will set it accordingly in each load command.

The fifth line is the *commit* command. This is either a command as would be entered via STOL or the string `NOCOMMIT`. If a command was entered, this command will be placed at the end of the formatted load so that it is uplinked after the load data.

The rest of the file is the load data, in hexadecimal notation. Load data lines optionally begin with `X`, and must contain an even number of hexadecimal characters.

9.2 Formatted Image Load Files

Formatted image load files look like this:

```
Table Load
98-133:05:40:05
 18 05 c0 00 00 09 00 01
 00 91 00 03 00 03 01 7e

 18 05 c0 00 00 cf 00 02
 00 00 00 00 00 c8 5f 38 b5 12 04 00 02 00 c9 99
 6f 42 f6 11 ee 3f 0f 87 30 74 a8 e3 d5 3f 7b 14
 ...
 32 33 3f 38 10 6b 09 3f 0a 00 00 00

 18 05 c0 00 00 07 00 04
 00 01 e0 c3 02 8c
```

They are uplinked via the LOADPKT directive. The first two non-blank and non-comment lines are ignored. Those lines usually are the load abstract and date copied from a load file compatible with the `load` directive.

10 Image Dumps

Image dump packets are captured automatically and written to a file by the Command Subsystem whenever a spacecraft table or memory dump command is issued by the user. The command subsystem determines if a command is a table or memory dump command by consulting the database: If the "dump flag" in the command definition is set to 'T' or 'M', then the command's "telemetry AppID for dump data" will contain the application ID of the telemetry packets which will carry the requested dump.

Spacecraft commands which abort dumps should be defined with the "dump flag" is set to 'A'. When ITOS sends such a command, it should abort dump packet collection. However, this currently doesn't work using the internal simulator; instead, when aborting test dumps, set the global mnemonic *GBL_DUMPCMDSTAT* variable to 0.

The command subsystem determines when it has received the complete image dump by examining information in the data field of each dump packet it receives, as described below. Dump collection also will time out after 60 seconds. If the information needed to determine when the dump is complete is not included in the dump packets, the timeout will terminate the dump.

If multiple copies of a dump are commanded and the appropriate mnemonics are defined for the dump packet, each copy will be collected individually.

Dump data are written to a file in ASCII hex format (see Section 10.2 [Image Dump Report Format], page 19). The filename can be specified using the STOL DUMPFILE directive, or will default to the name of the most recent LOAD file with the extension ".DMP.Cn" (where 'n' represents the copy number), or, if no LOAD was issued, will default to the name of the dump command with the extension ".DMP.Cn". **Existing files with the same name will be overwritten!**

When the dump is complete, the global mnemonic, *GBL_DUMPDONE* is set to the value 1. The global mnemonic, *GBL_DUMPFILE* is set to the name of the current dump file when the file is opened. The global mnemonic, *GBL_IMGDUMPDIR* specifies the directory where dump files are created.

10.1 Image Dump Packet Format

The database indicates which commands initiate memory or table dumps. See ⟨undefined⟩ [cmd], page ⟨undefined⟩, and which appid will contain the dumped data.

Memory dump packets may contain these mnemonics, all of which must be of type unsigned integer. If defined for the dump packets, they are used by the dump collection software as noted. Replace the *x* in the mnemonic names with an *M* for memory dumps and a *T* for table dumps.

CxDUMPCNUM

Tells to which copy of the memory dump this packet belongs. If this item is not defined for the dump packet, ITOS assumes that all dump packets belong to a single copy.

CxDUMPCTOT
: Gives the total number of copies which are being dumped. If this item is not defined for the dump packet, ITOS assumes one copy is being dumped.

CxDUMPENDADD
: Gives the address of the last byte in the memory dump. This is used to determine when the dump is complete: the dump is complete if *CxDUMPPKTADD* + *CxDUMPPKTSIZ* -1 >= *CxDUMPENDADD*, and all three of these items are defined for the dump packet.

CxDUMPPKTADD
: Gives the spacecraft memory address of the first byte of dump data in this packet. This item is used only to calculate when the dump is complete.

CxDUMPPKTSIZ
: Gives the number of bytes of dump data in this packet. If this item is not defined for the dump packet, the dump data is assumed to extend to the end of the packet.

CxDUMPSTADD
: Gives the spacecraft memory address of the first byte in the memory dump. If *cmdumpdata* is not defined for the dump packet, then the dump data begins 18 bytes past the offset of this item in the dump packet. If neither this item nor *cmdumpdata* is defined, the dump data begins 28 bytes from the start of the packet primary header.

CxDUMPDATA
: The offset of this item in the packet is the offset at which the dump data begins. Dump data continues through the end of the packet, or for *cmdumppktsiz* bytes. If this item is not defined for the dump packet, the dump data starts 18 bytes past the offset of the *cmdumpstadd* item.

Table dump packets also may contain these unsigned integer mnemonics, which are copied to the dump file header and used for no other purpose. They need be correct only in the first dump packet.

CTABLEID
: A table ID for the table being dumped.

CTMEMSOURCE
: A memory source ID for the table being dumped. This generally is used to distinguish among copies of a table in ROM, RAM, and other storage (usually other flavors of ROM or RAM).

10.2 Image Dump Report Format

The image dump packets are written to a file or group of files (if multiple dump copies were requested) in the format shown below. The first line of the file contains a description of the type of file (i.e. an "image dump file"). The second line of the file contains the date the file was created. The following lines beginning with the character # are comment lines.

For table dumps, there will be a comment line which identifies the table id and the memory source. Both table and memory dumps contain a comment line which indicates the copy number of the particular dump file and the starting address of the dump. The remaining lines which are prefixed with the character `X` contain the dump data extracted from the dump packets.

```
 Image Dump File
Date Created: 009-21:20:44
# Table ID: 95 (5fh)   Memory Source: 2
#
# Table Dump (Copy 1) Start Address: 0h
#
X 6a 16 35 9a 04 04 08 0a 00 0c 00 00 01 00 00 00
X 01 00 00 00 04 04 08 0a 00 0c 00 00 02 00 00 00
X 02 00 00 00 04 04 08 0a 00 0c 00 00 04 00 00 00
X 04 00 00 00 04 04 08 0a 00 0c 00 00 00 38 00 00
X 00 08 00 00 04 04 08 0a 00 0c 00 00 00 38 00 00
X 00 10 00 00 04 04 08 0a 00 0c 00 00 00 38 00 00
X 00 18 00 00 04 04 08 0a 00 0c 00 00 00 38 00 00
X 00 20 00 00 04 04 08 0a 00 0c 00 00 00 38 00 00
X 00 28 00 00 04 04 08 0a 00 0c 00 00 00 38 00 00
X 00 30 00 00 04 04 08 0a 00 0c 00 00 00 38 00 00
X 00 38 00 00 04 04 08 01 00 ec 00 00 00 04 00 00
X 00 00 00 00 04 04 08 0b 00 0e 00 00 00 10 00 00
X 00 00 00 00 06 04 08 0b
X 00 34 00 00 0f ff 00 00 07 c3 00 00 01 04 08 0b
X 00 36 00 00 0f ff 00 00 07 99 00 00 04 04 08 0b
X 00 0c 00 00 01 00 00 00 01 00 00 00 04 04 08 0b
X 00 0c 00 00 20 00 00 00 20 00 00 00 06 04 08 0b
X 00 3c 00 00 0f ff 00 00 07 d7 00 00 06 04 08 0b
X 00 3e 00 00 0f ff 00 00 07 d7 00 00 01 04 08 0b
X 00 0e 00 00 7f e0 00 00 4c e0 00 00 01 04 08 0b
X 00 0e 00 00 7f e0 00 00 4f 20 00 00 01 04 08 0b
X 00 0e 00 00 7f e0 00 00 59 a0 00 00 01 04 08 0b
X 00 0e 00 00 7f e0 00 00 7d 60 00 00 06 04 08 0b
X 00 0e 00 00 7f e0 00 00 7e c0 00 00 04 04 08 0b
X 00 0c 00 00 00 80 00 00 00 00 00 00 06 04 08 0b
X 00 66 00 00 0f ff 00 00
```

11 Image Load/Dump Verification

Once an image loads has been uplinked to the spacecraft, the user may issue a spacecraft dump command to dump the image just loaded. The dump image can be compared to the load image using the `VERIFY` directive (see ⟨undefined⟩ [VERIFY], page ⟨undefined⟩) to assure that the image was loaded correctly. The `VERIFY` directive causes a report to be generated which shows the results of a byte-byte comparison of the data in each file. Each byte that does not match will be marked with an *. An example of a verification report is shown below:

```
                        IMAGE COMPARISON REPORT

                            August  6, 1993


Load File Name: /home/tcw/loads/FMATSA001012.ATF  (105 bytes)
Dump File Name: /home/tcw/dumps/CTBLDUMP218-20:04:40.DMP  (105 bytes)

Number of Bytes Compared: 105
Number of Errors Found:   60


Addr   00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F
     ---------------------------------------------------------------
0000 | 2E  49  F6  A1  18  06  C0  00  00  09  00  01  01  00  00  00
     |*00 *00 *00 *00 *00 *00 *00  00  00 *00  00 *00 *00  00  00  00
0020 | 00  00  00  00  18  2E  49  F6  A2  18  09  C0  00  00  09  00
     | 00  00 *15  00 *00 *00 *00 *00 *00 *00 *00 *00  00  00 *00  00
0040 | 00  00  00  00  00  00  00  00  00  14  2E  49  F6  A3  18  03
     | 00  00  00  00  00  00  00 *15  00 *00 *00 *00 *00 *00 *00 *00
0060 | C0  00  00  09  00  05  05  00  00  00  00  00  00  00  1F  2E
     |*00  00  00 *00  00 *00 *00  00  00  00  00  00 *15  00 *00 *00
0100 | 49  F6  A4  18  03  C0  00  00  09  00  0C  04  00  00  00  00
     |*00 *00 *00 *00 *00 *00  00  00 *00  00 *00 *00  00  00  00  00
0120 | 00  00  00  10  2E  49  F6  A5  18  05  C0  00  00  09  00  02
     | 00 *15  00 *00 *00 *00 *00 *00 *00 *00 *00  00  00 *00  00 *00
0140 | 00  00  01  BC  41  00  00  00  E1
     | 00  00 *00 *00 *00  00 *15  00 *00
```

11.1 Command Checksums

There are a number of checksum routines available in ITOS that can be applied to commands. Most of the routines are CCSDS specific or specific to a mission's command format. Care must be taken not to specify a checksum routine that is inconsistent with the command type for which it is to be applied.

There are three levels within the command building process for which checksums can be applied:

11.2 Individual Commands

The inner most level of a checksum is applied to a command packet on a per command basis by specifying the name of the checksum within the ITOS database CMD definition record for each command for which the checksum is to be applied. The checksum is applied immediately after the command has been built as described by the database cmd record. Additionally a command checksum field can be defined and it's name can be passed as a parameter to the checksum routine name. If no checksum field name parameter is specified, the default name of LCLCHECKSUM is used. If the LCLCHECKSUM field does not exist to give direction otherwise, the checksum value will be appended to the end of the packet. See the CMD database record description for more detail.

11.3 All Command Packets

A single checksum can be applied to all command packets by setting the global mnemonic GBL_CMD_CHKSM_PKT to the name of a valid packet level checksum routine. Additionally, a checksum field called GBLCHECKSUM may be defined that the checksum routine can reference for instruction on how to insert the calculated checksum value into the command packet. The field must be specified as part of the special local header command. For example, to define a one-byte checksum field beginning at byte offset 7 of every command:

```
FLD|GBL_LCLHDR |GBLCHECKSUM|+|UI||7|0|8||||ZERO|
   "<HTML>
    Global checksum field appearing in all commands"
SUB|ZERO|default|+|0|N|Initialize to zero
```

If no checksum field is specified, the calculated checksum will be appended to the end of the packet (unless the checksum algorithm is documented otherwise).

The packet level checksum, if used, will be applied after any command-specific level checksum has been applied and is valid for both CCSDS and RAW command types (see CMD database record description). Again, the checksum routine specified must be consistent with the command type.

11.4 All Command Frames

A single checksum can be applied to all commands at the CCSDS Transfer Frame level by setting the global mnemonic GBL_CMD_CHKSM_TF to the name of a valid CCSDS Transfer Frame level checksum routine. This level of checksum will be applied after any command-specific and/or command packet level checksum have been applied.

12 Command Simulator

Table of Contents